VBA: Build & Calibrate Routes From Centerlines

Contributed by Bert Granberg
01, Jan. 2008
Last Updated 03, Jun. 2009

This code builds routes from centerline roads features that carry RT_NAME and RT_PART attributes.

It requires a set of calibration end points in a separate feature class that carry the m coordinate (measure) for end-point only calibration of the resulting route. REF_VALUE field carries the calibration values.

Assumes output location will be a file-based geodatabase and a projected (ie. UTM) coordinate system.

```
Option Explicit
Public Sub BuildRoutesParts()
   Dim outPath As String
   Dim calibrationLayerIndex As Integer
   Dim roadsLayerIndex As Integer
   Dim pCalibrationValueFieldName As String
   Dim calPtSearchTol As Double

   outPath = "c:/LRS/LRSOutput.gdb"
   calibrationLayerIndex = 0
   roadsLayerIndex = 1
   pCalibrationValueFieldName = "REF_VALUE"
   calPtSearchTol = 10 'meters

   Dim pMxdoc As IMxDocument
   Dim pMap As IMap
   Dim pCalibrationLayer As IFeatureLayer
   Dim pRoadsLayer As IFeatureLayer
   Dim pRoadsFC As IFeatureClass
   Dim pOutWS As IFeatureWorkspace
   Dim pOutFields As IFields
   Dim pOutFC As IFeatureClass
   Dim pGDS As IGeoDataset
   Dim pOutSR As ISpatialReference
   Dim dateStamp As String

   Set pMxdoc = ThisDocument
   Set pMap = pMxdoc.FocusMap
   Set pCalibrationLayer = pMap.Layer(calibrationLayerIndex)
   Set pGDS = pCalibrationLayer.FeatureClass
   Set pRoadsLayer = pMap.Layer(roadsLayerIndex)
   Set pRoadsFC = pRoadsLayer.FeatureClass
   Set pOutSR = pGDS.SpatialReference

   dateStamp = Format(Now, "yyyymmddhhmmss")
   Set pOutWS = openFGDBWS(outPath)
   Set pOutFields = createRouteFields(esriGeometryPolyline, pOutSR, True)
   Set pOutFC = createRouteFeatureClass(pOutWS, "Routes" & dateStamp, esriFTSimple, esriGeometryPolyline,
pOutFields)

   Dim csvRoutePartList As String
   Dim routePartList() As String
   Dim p As Long
   Dim routePartQStr As String
   Dim pRoutePartPolyline As IPolyline

   'Build Route Part List
   csvRoutePartList = getUniqueValues(pCalibrationLayer, "LABEL")
   routePartList = Split(csvRoutePartList, ",")
```

```
For p = 0 To UBound(routePartList)
    'Debug.Print routePartList(p)

    If routePartList(p) = "0302P_1" Then
        Debug.Print "here"
    End If

    routePartQStr = "DOT_RTNAME = '" & Left(routePartList(p), 5) & "' and DOT_RTPART = " _
                & Mid(routePartList(p), 7)
    Set pRoutePartPolyline = buildRoutePartPolyline(routePartQStr, pRoadsFC)

    If Not pRoutePartPolyline.IsEmpty Then
        Debug.Print routePartList(p) & ": " & pRoutePartPolyline.Length

        'check for calibration end points
        Dim pSpatialFilter As ISpatialFilter
        Dim pEndPtTopOp As ITopologicalOperator
        Dim pEndPtBuffer As IPolygon
        Dim pCalPtFCursor As IFeatureCursor
        Dim pCalPtFeature As IFeature
        Dim pPoint1 As IPoint
        Dim pPoint2 As IPoint
        Dim point1M As Double
        Dim point2M As Double
        Dim ptempPoint As IPoint
        Dim tempPointM As Double
        Dim fromPointError As Boolean
        Dim toPointError As Boolean
        Dim pMSegmentation As IMSegmentation
        Dim pOutFeature As IFeature
        Dim pMAware As IMAware
        Dim resultsCount As Long

        'Find polyline FROMPOINT's corresponding calibration point
        fromPointError = False
        Set pEndPtTopOp = pRoutePartPolyline.FromPoint
        Set pEndPtBuffer = pEndPtTopOp.Buffer(calPtSearchTol)
        Set pSpatialFilter = New SpatialFilter
        pSpatialFilter.WhereClause = "LABEL = '" & routePartList(p) & "'"
        Set pSpatialFilter.Geometry = pEndPtBuffer
        pSpatialFilter.SpatialRel = esriSpatialRelIntersects

        Set pCalPtFCursor = pCalibrationLayer.Search(pSpatialFilter, True)
        Set pCalPtFeature = pCalPtFCursor.NextFeature
        resultsCount = 0

        Do Until pCalPtFeature Is Nothing
            resultsCount = resultsCount + 1
            If resultsCount = 1 Then
                Set pPoint1 = pCalPtFeature.ShapeCopy
                point1M = pCalPtFeature.value(pCalPtFeature.Fields.FindField(pCalibrationValueFieldName))
            ElseIf resultsCount > 1 Then
                Debug.Print "too many end calibration points for " & routePartList(p) & " composite polyline FROM POINT"
                fromPointError = True
            End If
            Set pCalPtFeature = pCalPtFCursor.NextFeature
        Loop

        If resultsCount = 0 Then
            Debug.Print "too few end calibration points for " & routePartList(p) & " composite polyline FROM POINT"
            fromPointError = True
        End If


        'Find polyline TOPOINT's corresponding calibration point
        toPointError = False
        Set pEndPtTopOp = pRoutePartPolyline.ToPoint
        Set pEndPtBuffer = pEndPtTopOp.Buffer(calPtSearchTol)
        Set pSpatialFilter = New SpatialFilter
        pSpatialFilter.WhereClause = "LABEL = '" & routePartList(p) & "'"
        Set pSpatialFilter.Geometry = pEndPtBuffer
        pSpatialFilter.SpatialRel = esriSpatialRelContains

        Set pCalPtFCursor = pCalibrationLayer.Search(pSpatialFilter, True)
        Set pCalPtFeature = pCalPtFCursor.NextFeature
        resultsCount = 0

        Do Until pCalPtFeature Is Nothing
          resultsCount = resultsCount + 1
          If resultsCount = 1 Then
            Set pPoint2 = pCalPtFeature.ShapeCopy
            point2M = pCalPtFeature.value(pCalPtFeature.Fields.FindField(pCalibrationValueFieldName))
          ElseIf resultsCount > 1 Then
            Debug.Print "too many end calibration points for " & routePartList(p) & " composite polyline TO POINT"
            toPointError = True
          End If
          Set pCalPtFeature = pCalPtFCursor.NextFeature
        Loop

        If resultsCount = 0 Then
          Debug.Print "too few end calibration points for " & routePartList(p) & " composite polyline TO POINT"
          toPointError = True
        End If

        If Not (fromPointError Or toPointError) Then
          'flip routepart polyline?
          If point1M > point2M Then
            Set ptempPoint = pPoint1
            Set pPoint1 = pPoint2
            Set pPoint2 = ptempPoint
            tempPointM = point1M
            point1M = point2M
            point2M = tempPointM
          End If

          If (Round(pRoutePartPolyline.FromPoint.x * calPtSearchTol) / calPtSearchTol) _
            <> (Round(pPoint1.x * calPtSearchTol) / calPtSearchTol) Or _
           (Round(pRoutePartPolyline.FromPoint.y * calPtSearchTol) / calPtSearchTol) _
            <> (Round(pPoint1.y * calPtSearchTol) / calPtSearchTol) Then
            pRoutePartPolyline.ReverseOrientation
          End If

          Set pMAware = pRoutePartPolyline
          pMAware.MAware = True
          Set pMSegmentation = pRoutePartPolyline
          pMSegmentation.SetAndInterpolateMsBetween point1M, point2M

        End If
        Set pOutFeature = pOutFC.CreateFeature
        With pOutFeature
          Set .Shape = pRoutePartPolyline
          .value(pOutFeature.Fields.FindField("LABEL")) = routePartList(p)
          .value(pOutFeature.Fields.FindField("RT_NAME")) = Left(routePartList(p), 4)
          .value(pOutFeature.Fields.FindField("RT_DIR")) = Mid(routePartList(p), 5, 1)
          .value(pOutFeature.Fields.FindField("RT_PART")) = Mid(routePartList(p), 7)
          .value(pOutFeature.Fields.FindField("EFF_DATE")) = Now
          .Store
        End With
      End If
    Next p
```

```
End Sub

Public Function createRouteFeatureClass(featWorkspace As IFeatureWorkspace, _
                                        Name As String, _
                                        featType As esriFeatureType, _
                                        geomType As esriGeometryType, _
                                        pFields As IFields _
                                        ) As IFeatureClass

  On Error GoTo EH

  Set createRouteFeatureClass = Nothing
  If featWorkspace Is Nothing Then Exit Function
  If Name = "" Then Exit Function

  Dim pCLSID As UID
  Set pCLSID = Nothing
  Set pCLSID = New UID

    '' determine the appropriate geometry type corresponding the the feature type
  Select Case featType
      Case esriFTSimple
        pCLSID.value = "esricore.Feature"
        If geomType = esriGeometryLine Then geomType = esriGeometryPolyline
      Case esriFTSimpleJunction
        geomType = esriGeometryPoint
        pCLSID.value = "esricore.SimpleJunctionFeature"
      Case esriFTComplexJunction
        pCLSID.value = "esricore.ComplexJunctionFeature"
      Case esriFTSimpleEdge
        geomType = esriGeometryPolyline
        pCLSID.value = "esricore.SimpleEdgeFeature"
      Case esriFTComplexEdge
        geomType = esriGeometryPolyline
        pCLSID.value = "esricore.ComplexEdgeFeature"
      Case esriFTAnnotation
        Exit Function
  End Select


  ' establish the class extension
  Dim pCLSEXT As UID
  Set pCLSEXT = Nothing

  ' locate the shape field
  Dim strShapeFld As String
  Dim j As Integer
  For j = 0 To pFields.FieldCount - 1
    If pFields.Field(j).Type = esriFieldTypeGeometry Then
      strShapeFld = pFields.Field(j).Name
    End If
  Next

  Set createRouteFeatureClass = featWorkspace.CreateFeatureClass(Name, pFields, pCLSID, _
                            pCLSEXT, featType, strShapeFld, "")

  Exit Function
EH:
    MsgBox Err.Description, vbInformation, "createWorkspaceFeatureClass"
End Function
Public Function createRouteFields(geomType As Long, pSR As ISpatialReference, _
                              hasM As Boolean) As IFields
Dim pField As IField
Dim pFields As IFields
Dim pFieldEdit As IFieldEdit
Dim pFieldsEdit As IFieldsEdit
Dim hasmcoord As Boolean


'Create new Fields collection
Set pFields = New Fields
Set pFieldsEdit = pFields
'pFieldsEdit.FieldCount = 1

''
'' create the geometry field
''
Dim pGeomDef As IGeometryDef
Set pGeomDef = New GeometryDef
Dim pGeomDefEdit As IGeometryDefEdit
Set pGeomDefEdit = pGeomDef

' assign the spatial reference
'Dim pSR As ISpatialReference
If pSR Is Nothing Then
    Set pSR = New UnknownCoordinateSystem
    pSR.SetFalseOriginAndUnits 0, 0, 100
End If

pSR.SetMFalseOriginAndUnits -100000, 1000

If Not hasM Then
    hasmcoord = False
Else
    hasmcoord = True
End If

'' assign the geometry definiton properties.
With pGeomDefEdit
  .GeometryType = geomType
  .GridCount = 3
  .GridSize(0) = 1000
  .GridSize(1) = 10000
  .GridSize(2) = 100000
  .AvgNumPoints = 200
  .hasM = hasmcoord
  .HasZ = False
  Set .SpatialReference = pSR
End With

Set pField = New Field
Set pFieldEdit = pField

pFieldEdit.Name = "Shape"
pFieldEdit.Type = esriFieldTypeGeometry
Set pFieldEdit.GeometryDef = pGeomDef
pFieldsEdit.AddField pField


'Create Object ID Field
Set pField = New Field
Set pFieldEdit = pField

With pFieldEdit
    .Name = "OBJECTID"
    .AliasName = "FID"
    .Type = esriFieldTypeOID
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 10
    .Name = "LABEL"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 4
    .Name = "RT_NAME"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Length = 1
    .Name = "RT_DIR"
    .Type = esriFieldTypeString
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "RT_PART"
    .Type = esriFieldTypeSmallInteger
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "RT_DIR_ID"
    .Type = esriFieldTypeInteger
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "EFF_DATE"
    .Type = esriFieldTypeDate
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
With pFieldEdit
    .Name = "DEP_DATE"
    .Type = esriFieldTypeDate
End With
pFieldsEdit.AddField pField

Set pField = New Field
Set pFieldEdit = pField
```

```
    With pFieldEdit
        .Length = 100
        .Name = "EFF_NOTES"
        .Type = esriFieldTypeString
    End With
    pFieldsEdit.AddField pField

    Set pField = New Field
    Set pFieldEdit = pField
    With pFieldEdit
        .Length = 100
        .Name = "DEP_NOTES"
        .Type = esriFieldTypeString
    End With
    pFieldsEdit.AddField pField

    Set pFields = pFieldsEdit

    Set createRouteFields = pFields

End Function

Public Function openFGDBWS(inPath As String) As IFeatureWorkspace
        Dim pFGDBWSFactory As IWorkspaceFactory
        Set pFGDBWSFactory = New esriDataSourcesGDB.FileGDBWorkspaceFactory
        Set openFGDBWS = pFGDBWSFactory.OpenFromFile(inPath, 0)
End Function

Public Function getUniqueValues(inFL As IFeatureLayer, sFieldName As String) As String
      Dim pData As esriGeoDatabase.IDataStatistics
      Dim pCursor As esriGeoDatabase.ICursor
      Dim pStatResults As esriSystem.IStatisticsResults

      Set pCursor = inFL.Search(Nothing, False)

      Set pData = New esriGeoDatabase.DataStatistics
      pData.Field = sFieldName
      Set pData.Cursor = pCursor

      Dim pEnumVar As esriSystem.IEnumVariantSimple, value As Variant
      Dim iCnt As Integer
      iCnt = 0
      Set pEnumVar = pData.UniqueValues
      value = pEnumVar.Next

      Do Until IsEmpty(value)
        If iCnt = 0 Then
            getUniqueValues = value
        Else
            getUniqueValues = getUniqueValues + "," & value
        End If
        value = pEnumVar.Next
        iCnt = iCnt + 1
      Loop
End Function


Public Function buildRoutePartPolyline(inRoutePartQueryString As String, inRoadFC As IFeatureClass) As IPolyline

    Dim pFCursor As IFeatureCursor
    Dim pfeature As IFeature
    Dim pQF As IQueryFilter
    Dim pInGC As IGeometryCollection
    Dim pOutGC As IGeometryCollection
    Dim pGeometry As IGeometry
    Dim g As Long

    Set pQF = New QueryFilter
    pQF.WhereClause = inRoutePartQueryString
    Set pFCursor = inRoadFC.Search(pQF, True)
    Set pfeature = pFCursor.NextFeature
    Set pOutGC = New Polyline

    Do Until pfeature Is Nothing

        Set pGeometry = pfeature.ShapeCopy
        Set pInGC = pGeometry
        For g = 0 To pInGC.GeometryCount - 1
            pOutGC.AddGeometry pInGC.Geometry(g)
        Next g
        Set pfeature = pFCursor.NextFeature
    Loop

    pOutGC.GeometriesChanged
    Set buildRoutePartPolyline = pOutGC
    buildRoutePartPolyline.SimplifyNetwork
    Debug.Print inRoutePartQueryString & ": From X: " & buildRoutePartPolyline.FromPoint.x & " From Y: " &
buildRoutePartPolyline.FromPoint.y
    Debug.Print inRoutePartQueryString & ": To X: " & buildRoutePartPolyline.ToPoint.x & " To Y: " &
buildRoutePartPolyline.ToPoint.y

End FunctionPublic Sub flipRoutePartsToMatchMCoordDirection()

    Dim pMxdoc As IMxDocument
    Dim pMap As IMap
    Dim pCalibrationLayer As IFeatureLayer
    Dim pRouteLayer As IFeatureLayer
    Dim pRouteFC As IFeatureClass
    Dim pRouteGC As IGeometryCollection
    Dim pFCursor As IFeatureCursor
    Dim pFeature As IFeature
    Dim x As Integer
    Dim pCurve As ICurve
    Dim pNewGC As IGeometryCollection
    Dim reOrder As Boolean

    Set pMxdoc = ThisDocument
    Set pMap = pMxdoc.FocusMap
    Set pRouteLayer = pMap.Layer(2)
    Set pRouteFC = pRouteLayer.FeatureClass

    Set pFCursor = pRouteFC.Search(Nothing, True)
    Set pFeature = pFCursor.NextFeature

    Do Until pFeature Is Nothing

        reOrder = False
        Set pRouteGC = pFeature.ShapeCopy
        Set pNewGC = New Polyline

        For x = 0 To pRouteGC.GeometryCount - 1
            Set pCurve = pRouteGC.Geometry(x)
            If pCurve.FromPoint.M > pCurve.ToPoint.M Then
                Debug.Print pFeature.value(9) & "    oid(part): & "; pFeature.OID & "(" & x & ")     fromM:" & pCurve.FromPoint.M & "    to
pCurve.ToPoint.M
                pCurve.ReverseOrientation
                reOrder = True
            End If
            pNewGC.AddGeometry pCurve
        Next x
        If reOrder Then
            Set pFeature.Shape = pNewGC
            pFeature.Store
        End If

        Set pFeature = pFCursor.NextFeature
    Loop

End Sub
```